Path following algorithm for highly redundant manipulators

Erdinc Sahin Conkur
Mechanical Engineering Department, Pamukkale University, Denizli 20017, Turkey

Received 8 December 2002; received in revised form 1 April 2003

Abstract
An algorithm for path planning for highly redundant manipulators is presented in this paper. Assuming an approximated path is given by B-spline curves, path following is defined by requiring manipulator links to remain approximately tangent to these curves. The algorithm decouples manipulator links and establishes each link’s position relative to the curve using a numerical approach. As a result, the whole link achieves manoeuvring around the curve. A robust propagation procedure between links is also performed, which checks every link’s position and moves the links coming after the most recently moved link. These features lead to handling a huge number of degree of freedom (DOF) while keeping tight manoeuvring ability of highly redundant manipulators as illustrated with several examples.

Keywords: Highly redundant manipulators; Motion planning; Obstacle avoidance

1. Introduction
Redundant manipulators are defined as having an infinite number of solutions to the joint variables. They have the ability of choice of different sets of sequences of configurations as they come across interior or exterior constraints [10]. There are various potential applications for redundant or highly redundant manipulators, especially in applications where man-equivalent manipulative capabilities are required [29,37]. It is important to develop sophisticated motion planning algorithms to be able to utilise the advantages of redundant manipulators.

Motion planning problem for redundant manipulators is often considered as high level and low level planning [3]. If the path of an end-effector in a W-space (work space) is already given, the computation of a feasible joint path sequence for a redundant manipulator is referred to as redundancy resolution and often includes the computation of a generalised inverse of the Jacobian matrix. On the other hand, when the task is given as a specified goal point that the end-effector is to reach, the process of competing a feasible joint path sequence is referred to as the path planning problem [34].

1.1. Redundancy resolution
Regarding obstacle avoidance, there are two basic techniques to solve redundancy; the gradient projection and extended Jacobian matrix techniques. The gradient projection technique produces joint variables avoiding obstacles while the end-effector velocity is inserted using, e.g., a joystick [25]. A variety of performance criteria can be performed using the null space which controls only self-motion of the manipulator while the end-effector motion is obtained through the least norm solution [5,36].

The extended Jacobian technique transforms an underdetermined system to a determined one by...
means of additional constraints defined for the given task until the relationship between joint and task spaces, the Jacobian matrix, becomes non-redundant [30]. Additional kinematic constraints for obstacle avoidance are defined in several papers [2,4,6,35]. Although the idea of having a square Jacobian matrix is very attractive, augmented matrix singularities exist and there is no systematic way of choosing the constrained task functions. Both the gradient projection and extended Jacobian techniques are local techniques, which yields feasible solutions when the range of motion is small. There are few techniques that address global redundancy resolution such as [29,34]. Apart from these main approaches, Schilling et al. [33] give an example using resolved motion rate control equations for highly redundant manipulators, which achieves obstacle avoidance by making manipulator links approximately tangent to a given path.

1.2. Path planning for redundant manipulators

The basic motion planning problem may be formulated as follows.

Given an initial position and orientation and goal position and orientation of a robot with a number of degrees of freedom (DOF) in a W-space with obstacles, find a path that specifies a continuous sequence of positions and orientations of the robot between initial and goal configurations [21].

The gradient projection or extended Jacobian techniques can quickly react changes in the environment, whereas the level of competence may not always meet requirements of path planning. On the other hand, the objective of geometric motion planning for redundant manipulators is to generate a collision free path for the whole manipulator. From this point of view, the role of low level planning is reduced to carrying out basic operations determined by high level planning at the possible cost of sacrificing real time implementations [20].

Motion planning algorithms are based on a few general approaches such as roadmaps, cell decomposition, and potential field [21]. These approaches can be implemented in either W-space or C-space (configuration space). While W-space represents a Euclidean space where the robot moves, C-space represents all the configurations of the robot. Path planning for a robot in W-space is reduced to path planning for a point in C-space [18].

In the roadmaps approach, the connectivity of the free space is captured in the way that the set of feasible motions is mapped onto a network of one-dimensional lines. Roadmaps are constructed based on different methods such as visibility graph, Voronoi diagrams, freeway nets and silhouettes [15,21]. They usually include a huge number of nodes causing inefficiencies. Using roadmaps, there are several path planning algorithms in C-space for redundant manipulators; these algorithms are based on randomly processing C-space [1,37].

The cell decomposition method decomposes the free space into cells and constructs the connectivity graph presenting the adjacency among the cells. This graph is searched and a sequence of the cells which connect the start and goal points are obtained [17].

The potential field method has been of interest to resolve path planning and obstacle avoidance for mobile robots and manipulators since it was first introduced by Khatib [20]. The method uses a W-space under the influence of an artificial potential field where obstacles are represented by repulsive surfaces while the goal is represented by an attractive pole. Certain control points are identified on each link of the manipulator and the potentials of these points are combined to make the whole link avoid obstacles. One major criticism of the potential field technique is the local minima that are defined as the points in which the robot gets trapped before reaching the goal point [19]. The advantage of the local potential field approach is that it combines path planning, trajectory planning and control steps to one step in real time [32]. However, when the number of obstacles increases, oscillation occurs between obstacles and it becomes impossible to get close to an obstacle [12]. Another problem is that while the robot approaches obstacles, it is assumed that obstacle would exert an arbitrarily large repulsive potential [32]. Although the potential field technique is mostly considered as a local technique, it is possible to apply it globally using numerical potential fields or navigation functions that can be defined on the grid without local minima [13].

It seems that a systematic classification of path planning algorithms for redundant manipulators is missing or it is difficult to classify them due to the diversity of the algorithms presented. There are some
algorithms that are somewhat dependent on well-known general approaches for a global path, but different techniques are used for the path planning of manipulator links rather than the end-effector. For example, Reznik and Lumelsky [31] consider kinematic control using sensory data allowing soft contact with obstacles. Chirikjian and Burdick [8] develop a control model for highly redundant truss manipulators. This control model is based on a backbone curve used to determine the manipulator configuration through a fitting procedure. An alternative backbone curve formulation is presented in [11]. Mochiyama [26] derives new kinematic equations to fit a highly redundant manipulator onto a specified curve. Li and Trabia [22] state the path planning problem for highly redundant manipulators as a sequence of minimisation problems including penalty functions. The control of an infinitely flexible manipulator is proposed in [9] using Catmull-Rom splines. Ma and Konno [24] define a posture space that has lower dimension than C-space. A highly redundant manipulator design is presented in [23] using the posture space concept. Ellipsoids are used for the control of continuum robots in [14].

A path planning algorithm for highly redundant manipulators is proposed in this paper. Path planning is achieved by making manipulator links approximately tangent to a given path approximated by B-spline curves. Minimum distances of the end points of the link to the corresponding curve are determined by means of a numerical approach. Using these values, a number of control points on the link and their distances from the curve to the link are calculated. The most appropriate control point is selected and the link is moved accordingly. At the same time, a propagation procedure checks positions of all links. Therefore, the algorithm achieves to control a huge number of DOF while keeping tight manoeuvring ability of highly redundant manipulators.

This paper is organised as follows. Along with simple path following algorithms, B-spline curves and the minimum distance from one end of a link to the corresponding curve are explained in Section 2. Path following and the relationship between manipulator links and curves using minimum distances are defined in Section 3. The modes, same, opposite and switch, and the propagation procedure are also dealt with in this section. A number of computer simulations of the manipulators with links that range from 30 to 70 while following two different paths with seven B-spline curves are included in Section 4. Some points about the algorithm are discussed in Section 5 while conclusions are drawn in Section 6.

2. Two simple path following algorithms

2.1. A simple algorithm for following straight lines

Straight line paths can be followed by highly redundant manipulators in a very simple way. The aim of presenting such an algorithm is to demonstrate the easiness of the approach. Consider the straight line path and the links intersecting this line in Fig. 1. The straight line is expressed by a parametric equation whose parameter is \( u \). Similarly, each manipulator link is expressed by a parametric equation whose parameter is \( t \). The control strategy is simply to decide the direction of the link by specifying the parameter value between the points \( a(a_x, a_y) \) and \( e(e_x, e_y) \). The parameter \( t \) takes values between 0 and 1. Depending on \( t \), the link turns around its joint clockwise or counter-clockwise. Thus, the most distal link starts to become parallel to the straight line.

![Fig. 1. Following a straight line.](image-url)
The snapshots of an example are given in Fig. 2(a)–(c). The manipulator starts to move and finally reaches the goal point. The area the manipulator sweeps during execution is shown in Fig. 3. The angular change $\Delta \theta$ may also be chosen differently for each link such that the area swept becomes narrower while approaching the goal point. This can be done by adding higher angular change values to distal link variables.

The algorithm is quite capable of handling a high number of DOF even in its current form as seen in the example given in Fig. 4; a 15 link planar manipulator follows a straight line path and reaches the goal point successfully.

Although the algorithm presented in this section is for straight line paths, it also works for slightly curved paths. Fig. 5(a)–(c) shows an example of such an arrangement; a 6 link planar manipulator follows a slightly curved path.

2.2. A simple algorithm for following B-spline curves

The key question to path planning considered here is how to establish the relationship between links and curves. It may be concluded from the discussion so far that it is possible to develop a sophisticated algorithm for path planning using intersections between given curves and manipulator links. Yet, there are problems that would make it unfeasible to develop such an algorithm; it would be dependent on the degree of the curve function used, which means having a few roots to choose out of. Apart from the intersection case, there would be cases that seem to be difficult to deal with. The worst problem comes out in 3D. Since intersections are rare cases in 3D, manipulator links cannot be modelled by line segments. Instead, they may be extended sideways and modelled by square columns at the possible cost of computational complexity. For these reasons, a different strategy of interpreting the relationship between curves and manipulator links would be adopted. Before going into details of this strategy, B-spline curves used to approximate a given path are summarised below.
2.2.1. B-spline curves

The general formulation of open B-spline curves is given in [27] as follows:

\[ p_i(u) = U_k M_k P_k, \quad i \in [1 : n + 2 - k], \quad (1) \]

where

\[ U_k = \begin{bmatrix} u_{k-1} & u_{k-2} & \cdots & u_1 \end{bmatrix}, \]

\[ M_3 = \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \]

for \( k = 3 \), \( n \) is the number of control points, and \( P_k = [p_{j}] \quad j \in [i - 1 : i + k - 2] \) for open curves.

All B-spline curves used to approximate a given global path in this section take \( k = 3 \). For example, for \( n = 5 \) and, \( i \in [1:4] \) constitutes four B-spline curves that are expressed as follows:

\[ p_1(u) = U_3 M_3 [p_0, p_1, p_2], \quad p_2(u) = U_3 M_3 [p_1, p_2, p_3], \]
\[ p_3(u) = U_3 M_3 [p_2, p_3, p_4], \quad p_4(u) = U_3 M_3 [p_3, p_4, p_5], \quad (2) \]

where \( U_3 = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \).

Fig. 5 shows some sample curves and their control points calculated and drawn by MATHCAD.

2.2.2. Minimum distance from a point to a curve

As mentioned earlier, path planning using intersections between given curves and manipulator links has a few disadvantages. A numerical approach is preferred to establish the relationship between links and curves. Using this approach the direction of the rotation of each link is determined before an intersection occurs.

For the minimum distance between the point \( b(b_x, b_y) \) on the distal end of the link \( l_1(t) \) and the corresponding curve \( p_i(u) \) (Fig. 7), \( n \) points separating the curve into a number of segments are specified. It is done by dividing the curve parameter \( u \) by the number \( n \) and by incrementing \( u \) up to \( n \). Then, the distance between \( b(b_x, b_y) \) and each point on the curve is calculated. The minimum distance to the curve from \( b(b_x, b_y) \) would be between two points on the curve whose distances are the minimum compared to the other points. For the link and curve shown in Fig. 7, these are the ones between the parameter values \( u_1 \) and \( u_2 \). Then, the parameter value between these two distances is further divided by \( n \). This procedure is performed until a required precision is reached. In some cases, two minimum distances can be the same or almost the same although they are far away from each other. It causes one of these distances to be mistakenly chosen as the minimum distance. To prevent this, the gap between these latest distances on the curve is measured. If this value is more than a minimum value, the algorithm continues to divide latest two values to obtain the minimum one.
Using the minimum distance to the curve from \( b(b_x, b_y) \), the direction of the rotation of the link is specified (Fig. 8). Having decided the direction, the link is rotated with angular change \( \Delta \theta \) determined using

\[
\Delta \theta = \frac{l_m}{l},
\]

where \( l_m \) is the minimum distance from \( b(b_x, b_y) \) and \( l \) the link length.

When a curve is not tightly curved regarding link lengths, the resulting configuration of the manipulator is such that two end points of the link are on the curve. This situation can be seen clearly from the example given in Fig. 9(a)–(c) where the ends points of the first four links are on the first curve. The second curve is very tight regarding the link lengths (Fig. 9(c)), which imposes a constraint on the links. The end points of the first link (the end-effector) are on the second curve, but the link occupies relatively more space than it does for the first curve.

Considering Fig. 9(a)–(e), in the case of a tight curve, it is easily seen that the vicinity of the curve...
must be used to allow manipulator links to manoeuvre. When the curve is not tightly curved and \( b(b_x, b_y) \) is located in the middle of the link rather than on the distal end, the links would be approximately tangent to the curve as seen in Fig. 9(f). In the case of tight manoeuvring, the links do not even get approximately tangent to the curve and configurations similar to the ones in Fig. 9(c) are obtained.

Fig. 7. Minimum distance to the curve from the point \( b(b_x, b_y) \).

Fig. 8. Control strategy of simple curve following algorithm.
The control point \( b(b_x, b_y) \) may also be chosen somewhere between the distal end and the middle point of the link. In that case, the link will cross the curve partially depending on the position of the point on the link.

3. A sophisticated algorithm for following B-spline curves: the path following algorithm

3.1. Problem statement

The algorithms in the previous sections are only able to follow straight or slightly curved paths. Perfect path following is achieved in the sense that manipulator links become co-linear to the straight line as can be seen from the examples already given. When the curvature of the curve is high and manipulator link lengths are relatively short, the area manipulator links sweep around the curve is reasonably small. On the other hand, relatively long link lengths with low curvature make it necessary to sweep more area around the curve. In such a case, manipulator links cut the curve in the middle if the algorithm presented in Section 2.2 is used. However, to achieve tight manoeuvring, it is necessary to get approximately tangent to curved paths.

Assume that a path created for point robots passes near obstacles rather than passing in the middle of obstacles when it sharply changes direction. Assume further that this path is approximated by B-spline curves. Then, when the path is tightly curved inwards (when...
it is convex) as was seen earlier in Fig. 9 (c)–(e) obstacles would be inside convex B-spline curves. This means that if manipulator links keep outside of these convex curve, they avoid obstacles. As mentioned before, they must sweep some area around the curve to be able to physically manoeuvre around it. Therefore, path following may be defined as follows.

An approximate path given by B-spline curves is said to be followed by a highly redundant manipulator if manipulator links approach convex curves from outside at each increment and remain approximately tangent to them without crossing them except for their end points.

As the expression except for their end points states that there are situations where manipulator links must cross curves. These are those of crossing two adjacent curves when the current link reaches the end of the current curve and needs to switch to the next curve.

Using just one control point as in the algorithms in the previous section will not achieve the objective stated above. Two control points may be used, but there is no criterion of choosing the most appropriate one. Having few control points always leaves room for collision regarding tight manoeuvring. This problem may be solved by determining the most appropriate control point for each increment, which will be considered in the next section.

3.2. Establishing relationship between links and curves

A geometric property should be found to specify angular changes for the joint variable $\theta$, which results in the link getting closer to the curve in the way that no part of the whole link must cross the curve. The minimum distance to the curve from the proximal and distal ends of the link has already been explored in Section 2.2.2. However, it does not give sufficient information on crossing curves. More information on crossing curves can be obtained using the parameter values of the curve at these points. To do so, the parameter value between the points $c(x_c, y_c)$ and $d(x_d, y_d)$ is divided by a number $n$ (Fig. 10). Starting from the parameter value $u_c$, it is incremented up to the param-

![Fig. 10. Relationship between the link and the curve.](image-url)
Line segments perpendicular to the link are drawn between the link and the curve. In the process of determining these line segments, any sense changes are also recorded as mentioned before. Since all the line segments are parallel to each other, a line segment only changes sense when the link crosses the curve. Regarding second degree B-spline curves used in this algorithm, there are two possible types of crossing the curve; first, the distal end of the link may cross the curve slightly, one or few line segments next to this end change sense (Fig. 11(a)). Second, the distal end of the link does not cross the curve, instead the link crosses the curve such that the link intersects the curve in two points and the length of the link between these two points becomes inside of the curve (Fig. 11(b)). The next step is to distinguish between these different situations, which may be stated as follows.

A manipulator link is said to be in the **same sense** mode when all line segments are in the same sense. Otherwise, it is said to be in the **opposite sense** mode. There is one more mode called the **switch mode**. A manipulator link is said to be in the **switch mode** when the distal end of a link reaches the end of the curve and starts to follow a straight line. Details of these modes are given as follows.

### 3.3.1. Same sense mode

Manipulator links are normally in this mode. The first task in this mode is to select the most appropriate line segment out of the line segments found in Section 3.2. The most appropriate line is the one that does not cause the link to cross the curve. It gives the information on the direction of the link rotating about the point \( a(x_l, y_l) \). To find the most appropriate line, first, similar triangles are formed out of each successive two lines and corresponding link lengths (Fig. 12(a) and (b)). As seen from Fig. 12(b) the only unknown length is \(|\mathbf{k}_{i}|\) that can easily be determined through similar triangles:

\[
|\mathbf{k}_{i}| = \frac{|\mathbf{a}_{i} \cdot \mathbf{j}_{h}|}{|\mathbf{a}_{h}|}.
\]  

(10)

A value \( g \) is defined as follows:

\[
g = |\mathbf{k}_{i}| - |\mathbf{s}_{i}|.
\]  

(11)

g must be negative to select a line segment as the most appropriate line segment. If \( g \) is positive, it means that...
the first line segment compared with the second one crosses the curve. Suppose that lines 7 and 8 are to be compared to decide which one is more appropriate (Fig. 12(a)). Line 7 crosses the curve. Since line 7 cannot be selected as the appropriate line segment, a comparison is made between line segments 8 and 9. This procedure continues until the most appropriate line segment is found. In the example in Fig. 12(a), it is the line segment numbered as 9 whose \( s \) value is negative.

Having found the most appropriate line segment, the remaining task is to find the value of the angular...
change $\Delta \theta$ for the joint variable $\theta$. The angular change $\Delta \theta$ can be calculated by dividing the line segment $l_m$ by the corresponding link length $l$. However, the algorithm does not use this value directly since the movements of the other links that come before this link cause it very often to cross the curve slightly when it is very close to the curve. Instead, a predetermined gap $l_s$ is left between the corresponding point of the link and

Fig. 12. (a) Finding the most appropriate line. (b) Details of finding the most appropriate line.
the curve. The value of $\Delta \theta$ is, therefore, calculated by

$$
\Delta \theta = k \frac{t_m - l_s}{l_l},
$$

(12)

where $k$ is a coefficient.

When the distance $l_m$ is greater than the distance $l_s$, the link approaches the curve up to the point $r(x, y)$ (Fig. 13(a)). When $l_m$ is less than $l_s$, the link moves away from the curve up to the point $r(x, y)$ (Fig. 13(b)). Consequently, the link is always kept outside of the curve.

The coefficient $k$ acts as a filter in the case of tight manoeuvring. If a link approaches the curve very closely as a result of the movements of the other links that come before it, $l_m$ would be less than $l_s$, the algorithm will attempt to move the link away from the curve. Depending on the configurations of the other links, this backward movement may occur repeatedly.
If only a percentage of $\Delta \theta$ is used to move the link toward the curve, this backward movement can be reduced. This is achieved by giving the coefficient $k$ a value between 0 and 1. Note however that this procedure does not need extra free space. It still uses the space around the tight manoeuvring area.

The direction of $\Delta \theta$ is simply determined through the difference between the link’s angle $\Delta \theta$ and the vector’s angle $\Delta \theta$ (Fig. 13(a) and (b)).

When tight manoeuvring is required, the most appropriate line segment would not be the last line segment close to the proximal end of the link. It would be somewhere between the ends of the link. On the other hand, when the curvature of the curve is very high, the last line segment would be selected. If this is the case, it is more suitable to take

$$l_m = |\text{bd}|,$$  \hspace{1cm} (13)

$$l_t = l.$$  \hspace{1cm} (14)

The direction change may become unstable since manipulator links at the initial stage are perpendicular or almost perpendicular to the tangent vector at corresponding points on the curve. As the point from which it is at minimum distance to the distal end of the link is used to determine the direction change, it is vitally important that this point on the curve is at the correct position at the initial stage. Otherwise, the link may start to move in the opposite direction. Fortunately, there is an easy reference to ensure that the right direction is chosen. This reference is the parameter of the curve since its value increases in the direction of the manipulator motion.

3.3.2. Opposite sense mode

In this mode the equation below is employed to obtain $\Delta \theta$:

$$\Delta \theta = \frac{|\text{bd}|}{l}$$  \hspace{1cm} (15)

for the first type of intersection of the link with the curve (Fig. 11(a)).

The second type of intersection is not included in the algorithm since the previous link automatically helps the current link recover from such a situation.

3.3.3. Switch mode

The same and opposite modes successfully achieve to produce $\Delta \theta$ values to follow curves. Nonetheless, they do not yield very impressive results when a link needs to switch from one curve to another. Hence, the link that reaches the end of the current curve starts to follow a line segment temporally until it switches to the next curve. This line segment is easily constructed using the second and third coefficients of the current curve equation. Considering Fig. 6, the link to be followed after the first curve is the one whose start and end points are $p_1$ and $p_2$. The link to be followed after the second curve is the one whose start and end points are $p_2$ and $p_3$ and so on. The link starts to follow the next curve when it is fully at the other side of the next curve while it follows the corresponding line. $\Delta \theta$ is specified simply by

$$\Delta \theta = \frac{l_m}{l_t}.$$  \hspace{1cm} (16)

3.3.4. Propagation of links

Once $\Delta \theta$ is determined for each link, the relationship between links should be established. It is usually not practical or useful to move all manipulator links at the initial stage especially in the case of a huge number of links. The initial positions of manipulator links are set in a concertina-like configuration. It is probably the most appropriate position for highly redundant manipulators rather than being an obligatory condition. The links that are already initialised are called the active links. The other links that wait to be initialised are called the passive links. At the initial stage, all of the links are passive except the most proximal link. Whenever a link’s distal end point becomes close enough to the curve, the link before this link is set active.

Angular changes produced by the equations presented in the previous section may be very high for stable motion. This issue is especially important both at the initial stage and at thigh manoeuvring. The angular change $\Delta \theta$ is exceptionally high at the initial stage since the distance of the distal end of the link to the curve is maximum or almost maximum. Moreover, since the equations for $\Delta \theta$ in the previous section are actually approximate, they can give sufficiently accurate results only when $\Delta \theta$ is relatively small. The angular changes produced at tight manoeuvring are not as high as it is at the initial stage, but still affect stability of the control of the manipulator. This problem is solved by imposing a limit on $\Delta \theta$. Each time when
the link is moved, it will not have an angular change
more than the limit value $\theta_{\text{lim}}$. Dividing $\Delta \theta$ by $\theta_{\text{lim}}$
gives the number of successive angular changes. At the
initial stage, before these successive angular changes
run out, the link may intersect the curve due to the
reason explained above. This problem is solved by re-
stricting the number of successive angular changes to
a few changes. When these changes run out, $\Delta \theta$ is
calculated again. When the minimum distance of the
distal end to the curve becomes relatively small, this
problem is solved automatically since much more ac-
curate angular changes are obtained. One more detail
is that the number of angular change values as a result
of dividing the angular change by $\theta_{\text{lim}}$ may produce
a remaining value which is less than the limit value.
This value is added the first increment. Finally, the
propagation procedure below is performed.
When $n$th active link is moved, the other links which
come after it are moved. If the angular change of any
of them, say $k$th link, is higher than four times the
limit value, before returning to $n$th link, $k$th link is
moved and the other links which come after it are
moved again obeying the rule for $n$th link, i.e., $n$,
then $n-1, n-2, n-1, n; \ldots; k+1, \ldots, n-2, n-1, n$.
The value four times above rather than one time is set
to reduce the computational complexity.

4. Computer simulations

A number of computer simulations using a P664
PC with 256 Mb RAM have been carried out to
verify the path following algorithm. Two paths are
given to be followed by three highly redundant ma-
nipulators. Each path consists of seven B-spline
curves whose control point co-ordinates are shown in
Table 1.

4.1. First path

4.1.1. 30 link planar manipulator, link lengths of 120

A 30 link planar manipulator at the initial pos-
tion and the first path to be followed are shown in
Fig. 14(a). The line segments derived from the curve
coefficients are also shown in Fig. 14(a). The most
proximal link’s angle $\theta$ on the base is 1.52 rad mea-
sured from the positive $x$-axis. Each link’s relative an-
gle measured from the previous link changes in order
as 3.05 and $-3.05$ rad. The link lengths are of 120.
The predetermined gap $l_{\text{p}}$ is set to 5.0 and the angular
change limit value $\theta_{\text{lim}}$ is set to 0.01 rad. The number
of control points is chosen as 10.

The snapshots (Fig. 14(a)–(e)) demonstrate that the
30 link planar manipulator follows a complex path that
consists of seven B-spline curves. The area that the
manipulator links sweep during the implementation of
the task can be seen in Fig. 14(f). The number of the
active links is 26 when the end of the last curve is
reached as is seen in Fig. 14(e). The running time for
this manipulator to reach the configuration shown in
Fig. 14(c) is 5 s and to reach the final configuration
shown in Fig. 14(e) is 20 s.

4.1.2. 30 link planar manipulator, link lengths of 80

The link lengths of the manipulator above are re-
duced to 80. Everything else remains the same. The
final configuration of the manipulator can be seen in
Fig. 15(a). The path becomes less tight regard-
ing the link lengths and, as it would be expected, the
area that the manipulator links sweep is reduced
(Fig. 15(b)).

4.1.3. 70 link planar manipulator, link lengths of 50

The link lengths of the manipulator are further re-
duced to 50. Also, the number of links is increased
to 70. Since the link lengths are significantly reduced,
the predetermined gap $l_{\text{p}}$ is set to 2.0 and the number
of control points are chosen as 4.

The number of active links is 55 when the task
is completed. As can be seen from Fig. 16(a), it is
very difficult to distinguish the curves and the links
except for places where tight manoeuvring is neces-

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Co-ordinates of the control points of the curves used in examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First path</td>
</tr>
<tr>
<td></td>
<td>$x$</td>
</tr>
<tr>
<td>$p_0$</td>
<td>0</td>
</tr>
<tr>
<td>$p_1$</td>
<td>500</td>
</tr>
<tr>
<td>$p_2$</td>
<td>1200</td>
</tr>
<tr>
<td>$p_3$</td>
<td>850</td>
</tr>
<tr>
<td>$p_4$</td>
<td>730</td>
</tr>
<tr>
<td>$p_5$</td>
<td>400</td>
</tr>
<tr>
<td>$p_6$</td>
<td>300</td>
</tr>
<tr>
<td>$p_7$</td>
<td>700</td>
</tr>
<tr>
<td>$p_8$</td>
<td>1400</td>
</tr>
</tbody>
</table>
The path becomes much less tight regarding link lengths and the area that the manipulator links sweep is much more reduced (Fig. 16(b)). The running time for this manipulator to reach the final configuration is 3 min.

4.2. Second path

4.2.1. 30 link planar manipulator; link lengths of 80

This path is not only a long path as the first one, but also designed to accommodate tight manoeuvring.
The first, second and fourth curves are very tight. The rest are loose. The most proximal link’s angle $\theta$ on the base is 0.9 radian measured from the positive $x$-axis. Each link’s relative angle measured from the previous link changes in order as 3.05 and $-3.05$ rad as above. The link lengths are of 80. The initial position of the manipulator and the first path as well as the line segments derived from the curve coefficients are shown in Fig. 17(a). The predetermined gap $l_s$ is set to 5.0 and the angular change limit value $\theta_{lim}$ is set to 0.01 rad. The number of control points is chosen as 10.

The 30 link planar manipulator follows this path which is more complex than the first one as seen from the snapshots (Fig. 17(a)–(g)). It can easily be observed that there is more gap than the first example between the curves and the links in places where tight manoeuvring is required. Curve following improves when the curve is loose. Consider also Fig. 17(b) that shows the area that the manipulator links sweep during the implementation of the task. All the links become active and contribute to path following. The running time for this manipulator to reach the configuration shown in Fig. 17(d) is 8 s and to reach the final configuration shown in Fig. 17(g) is 45 s.

4.2.2. 50 link planar manipulator, link lengths of 50

The link lengths of the manipulator above are reduced to 50. Everything else remains the same except for the number of control points which are chosen as...
Fig. 17. (a)–(g) Snapshots of 30 link planar manipulator, link lengths of 80, second path.
4. Fig. 18(a) shows the final configuration of the manipulator. The path becomes less tight regarding link lengths and the area that the manipulator links sweep is reduced (Fig. 18(b)). The number of active links is 47 when the task is completed.

4.2.3. 70 link planar manipulator, link lengths of 30

The link lengths of the manipulator are further reduced to 30. Also, the number of links is increased to 70. Although the link lengths are significantly reduced, the predetermined gap $l_s$ is still set to 5.0. This is because the angular change limit value $\theta_{lim}$ is set to 0.03 rather than 0.01 to speed up the execution process. The number of control points is chosen as 4.

All the links become active and the most proximal link settles on the first curve before the manipulator reaches the end of the last curve (Fig. 19(a)). It is difficult to distinguish between the curves and the links except for places where tight manoeuvring is necessary, and the area that the manipulator links sweep is reduced (Fig. 19(b)). The running time for this manipulator to reach the final configuration is 75 s.

5. Discussion

One question about the path following algorithm may be whether there is a pattern that manipulator links follow. As seen from the simulations, configurations show a regular pattern. Therefore, manipulator links configurations can be predicted no matter how long and how complex the path is. In places where the path is relatively less curved, link lengths simply become parallel to the path. On the other hand, in places...
where the path is relatively more curved, links occupy some area right after the path is tightly curved. This is where tight manoeuvring occurs and in that case it is important to put an upper limit on the tightness.

The path following algorithm not only achieves tight manoeuvring but also achieves handling many DOF as it is clear from the examples given. One reason for this is that the path following algorithm decouples links, determines each link movement individually and organises the relationship between all active links. Another reason is that each active link uses only one curve or line equation to follow. Whenever necessary, a manipulator link switches simply from one curve or line equation to another.

The running time is extremely short when each manipulator link is moved just once at each time increment as in the simple algorithms presented earlier. Nonetheless, whenever a link is moved, it is obligatory to check other links’ positions for complex paths. Therefore, a huge number of links and complex paths are the reasons for relatively long execution times. However, recall that all the execution times are obtained using a P664 PC with 256 MB RAM.

The path following algorithm automatically prevents self-intersection as well. The algorithm keeps manipulator links out of each curve. Manipulator links never cross the inside of curves. Therefore they cannot cross each other. The minimum distance between the links on the sides of a curve is limited by that curve’s equation.

There are few coefficients introduced in the previous sections. They must be adjusted before starting path following. Adjustments are easy as the coefficients are linearly proportional to the link lengths. Moreover, the algorithm works for a range of values for these coefficients, not just one value for a specified link length.

The path following algorithm can easily be extended to 3D. The easiness comes from the fact that intersections are not used when calculating minimum distances. Therefore, the complexity of dealing with higher orders is avoided. Calculation of a B-spline curve at a point in 3D is as simple as in 2D. Once the most appropriate point is chosen, the remaining task is to distribute angular change values for each DOF since each link would have two DOF in 3D.

Paths that compose of B-spline curves may be obtained from paths for point robots generated by harmonic potential fields. The path given for point robots passes near obstacles, the area on the side away from obstacles would be wider than the other. This side allows the manipulator to manoeuvre while obstacles are avoided.

One way of reducing the running time is to optimise the path following algorithm. For example, each time a link is moved, minimum distances from the ends of each link to the curves are calculated. Nevertheless, a link’s proximal end is the distal end of the next link. It means that once that link’s proximal end’s minimum distance is calculated, it is not necessary to recalculate it for the next link. It should save a considerable amount of time. Also, the whole curve is scanned when minimum distances are calculated, but this is necessary only for the first time. Once a point’s position is known, next time a curve parameter value plus and minus around this point could be chosen. It would make it faster to find the minimum distance of the point to the related curve. A further improvement can be made considering whether the curvature of the curve is very high or the curve is simply a straight line requiring just one control point.

Curve-fitting approach is an efficient way of path planning for highly redundant manipulators. Among several attempts made to reduce the dimension of C-space [15,16], one way is to define a backbone curve [23,28]. When a backbone curve is used, the C-space dimension is reduced to the number of the backbone curve variables. Apart from C-space, many other methods for point robots can be used to construct a curve for path planning as in [7,11]. Among these different backbone curve formulations, the path following algorithm presented in this paper uses a simple numerical algorithm to fit the manipulator on a given curve defined by B-spline curves. This algorithm leads to controlling a huge number of DOF while increasing obstacle avoidance ability of highly redundant manipulators. It is a global approach in nature. Compared to the paper by Schilling et al. [33], the path following algorithm does not suffer from singularities.

The design of highly redundant manipulators is divided into three main categories. The first one is called discrete morphologies that consist of a finite number of non-distributed actuators. Serial link manipulators considered in this category. The second one is continuous morphologies whose geometry can be deformed continuously due to actuation distributed through the
manipulator structure. The third one is cascades of parallel platform modules. The manipulator structure is constructed adding these modules to each other [7].

The method presented in this paper is developed for serial link manipulators and can easily be applied to them. It can also be applied to manipulators in the other categories mentioned above since having a large number of links resembles continuous morphologies as seen from the examples given.

6. Conclusions

The path following algorithm presented in this paper makes highly redundant manipulator links follow a given path approximated by B-spline curves. Path following is defined by requiring manipulator links to remain approximately tangent to these curves. The algorithm consists of three main steps; first, minimum distances of the end points of the link to the corresponding curve are determined. Using these values, a number of control points on the link and their distances from the curve to the link are calculated. Second, the most appropriate control point is selected and the link is moved accordingly. Third, a propagation procedure ensures to check positions of all links. Therefore, the algorithm achieves not only to control a huge number of DOF but also to keep tight manoeuvring ability of highly redundant manipulators. The algorithm is global in nature and contains no singularities. Several examples are included to show these features of the algorithm.

References


[19] F. Janabi-Sharifi, D. Vanke, Robot path planning by integration the artificial potential field approach with simulated annealing.


[38] E Sahin Conkur received the B.S. degree in Mechanical Engineering from Gazi University, Turkey, in 1988. Between 1989 and 1993, he was a lecturer with Erciyes University, Kayseri, Turkey. He received the Ph.D. degree in Mechanical Engineering from the University of Bristol, Bristol, UK, in 1998. He is an Assistant Professor of Mechanical Engineering at Pamukkale University, Denizli, Turkey. His research interests include control of redundant manipulators.